FastReact
In-Network Control and Caching for Industrial Control Networks using Programmable Data Planes

Authors:
Jonathan Vestin <jonathan.vestin@kau.se>
Andreas Kassler <andreas.kassler@kau.se>
Johan Åkerberg <johan.akerberg@se.abb.com>
Fusion of Compute and Data Center Networking

Next generation Programmable Switch Architecture
- Fully Programmable Match-Action Table
- Outsource simple tasks to programmable switches
- Network can act as a Distributed Data Store / Control

Packet Processing
- WxLAN
- NVGRE
- Your Own Protocol

Datacenter Network
Facilitated Through P4

New Generation of P4 Programmable Datacenter Fabric enables the Network dataplane to perform simple Compute Tasks.

Next Generation Compute Infrastructure

The P4 Switch

- Low overhead (high speed, ideally line rate)
- Define your own packet processing pipelines
- Improved signaling, monitoring and telemetry
- Change functionality with software upgrades as opposed to hardware changes
- Reduce complexity
Traditional Switch Architecture

Control Plane

Table Management

TCAM / Match Table

Ingress Packets

NPU/ASIC

Egress Packets

Routing Protocol

Data Plane

std.inport=1,ethdst=22:.. output:2
std.inport=2,ethdst=22:.. output:1
std.inport=1,ethdst=11:.. output:2
OpenFlow-based Switch

Dumb Control Plane

Data Plane

TCAM / Match Table

NPU/ASIC

Ingress Packets

Egress Packets

OpenFlow Controller

Policies/Signaling

Table Management

Control Signaling

Management
P4-based Switch

Dumb Control Plane

P4 Match-Action Tables

Compiled P4 Program

P4 Controller

P4 Compiler

P4 Program

Ingress Packets

Egress Packets
Many Applications

- **Silkroad: Layer-4 Load Balancer**
  Replace 100 servers or 10 dedicated boxes with one programmable switch.
  Track and maintain mapping for 5-10M HTTP flows.

- **NetCache: Cache for Key-Value store**
  Switch can act as an in-network cache, 1-2M op/sec

- **HULA: Dataplane load balancing**
  ToR-ToR Congestion tracking through programmable probes.
  Flowlet switching among least congested paths.
Both Software and Hardware Targets

- Software Switches (BMV2)
- P4-XDP Compiler
- P4-DPDK Compiler
- Agilio CX (Netronome)
- P4-NetFPGA (Using Xilinx NetFPGA-SUME)
- Tofino Switch (Barefoot Networks)
Industry 4.0: Highly Integrated Smart Production
Industry 4.0: Highly Integrated Smart Production

Smart Production
Multiple physical processes on shop floor.
○ In parallel, but highly dependant
○ Precision sensing provides massive amounts of data.
○ Control algorithms run in local cloud.

Challenges
○ High control requires stable and ultra-low latency.
○ Raw sensor data requires huge data rates.
(In particular imaging and AR)
Industry 4.0: Highly Integrated Smart Production

Local Decision Making
- Early reaction reduces time required for processing.
- Reduces network data rate
- Fewer devices that can fail

FastReact
- Implemented in P4 data plane programming language
- Sensor value history, moving averages, etc.
- Trigger local actions based on locally stored data

```python
if (sensor1 > 50) && (sensor2 < 25):
    trigger_actuator(<portno>)
```
FastReact: In-Network Monitoring, Caching and Control

Sensors
Actuators

Wired Network

Cloud Based Control
FastReact: In-Network Monitoring, Caching and Control

Wired Network

Cloud Based Control

Parser
- Ethernet
- IPv4
- ARP
- TCP
- UDP
- Sensor

Match/Action Tables
- ip.src == 10.0.0.3, forward(2)
- ip.src == 10.0.0.4, drop()

Packet Metadata

Egress Queues

FastReact P4 Program
FastReact: In-Network Monitoring, Caching and Control

Collect from multiple sensors

Also keep historical values

Switch Local Memory
Temp4C: 67, 66, 63, 65...
Temp4D: 33, 33, 33, 32...
Temp4E: 32, 32, 31, 32...

Can perform actions on certain conditions
if Temp4C > 70: notify actuator
FastReact: In-Network Monitoring, Caching and Control

Also supports OpenFlow BFD/Fast Failover-like recovery in case of equipment failure.
FastReact: In-Network Monitoring, Caching and Control

- Implemented entirely in P4_16 using registers to configure desired logic.
- Logical expressions must be stated in CNF, but controller could simplify.

### Ingress Pipeline

**Apply:**
- Record Sensor Info
- Run Decision Logic
- Apply Route Table

**Route:**
- **Match**
  - ingress_port: exact
  - hdr.sensorid: exact
  - hdr.pkttype: exact
  - hdr.sensor.isValid?: No

**Actions**
- forward(port)
- forward_mod(port, ip, mac, ...)
- send_up(port)
- drop

**Route Table**
- timestamp = \( \text{reg}(ts, egress\_port) \)
- port_up = (timestamp - pkt.ingress_timestamp > \text{reg}(timeout))

**Egress Pipeline (empty)**

### Parsers

- Parse Ethernet Metadata
- EtherType
- IPv4 Metadata
- Protocol
- Parse UDP Sensor
FastReact: In-Network Monitoring, Caching and Control

Prototype developed for the BMV2 target, and tested in simple experiment setup.
FastReact: FastReact vs Central Processing

![Graph showing sensor-actuator delay over time for FastReact and Reference systems. The graph compares the performance of FastReact with a reference system, indicating that FastReact has a lower sensor-actuator delay over time.](image-url)
FastReact: Failure Recovery

![Graph showing sensor-actuator delay over time for Actuator 1 and Actuator 2]
FastReact: Switch Logic

Sensor-Actuator Delay (ms)

Time (s)

Sensor 1A Value

Sensor 1B Value
FastReact: Integer Values

![Graph showing sensor values and actuator delays over time.](image-url)
Conclusion

FastReact works by moving part of the industrial control logic to the core or edge switch. This can reduce network latency and data usage.

Future Work

○ Implement on hardware switch.
○ Design controller application and CNF simplification.